```

// Function
// - fundamental building block in the program
// - subprogram can be used multiple times
// - performs a task or calculates a value

// 1. Function declaration
// function name(param1, param2) { body... return; }
// one function === one thing (한개의 함수는 한개의 일만 수행하도록 작성)
// function naming: doSomething, command, verb
// 함수는 이름을 세분화하여 작성하는 것이 좋음
// e.g. createCardAndPoint -> createCard, createPoint
// function is object in JS

function printHello() {
  console.log("Hello");
}
printHello();

/* 파라미터로 메세지를 전달 */
/* JS는 type이 없기 때문에 message가 string을 전하는지 number를 전하는지 모름 */
function log(message) {
  console.log(message);
}
log("Hi");
log(1234);
/* TS 예시 */
/* function log(message : string) : number {
  console.log(message);
  return 0;
} */

// 2. Parameters
// premitive parameters: passed by value
// object prarameters: passed by reference
function changerName(obj) {
  obj.name = "coder";
}
/* 오브젝트에 할당 시킴 -> 메모리에 레퍼런스가 저장됨 -> 레퍼런스가 오브젝트를 가리킴 */
const ellie = { name: "ellie"};
changerName(ellie);
console.log(ellie);

// 3. Default parameters (added in ES6)
function showMessage(message, from = "unknown") {
  /* if(from === undefined) {
    from = "unknown"
  } */ /* -> 예전방식 */
  console.log(`${message} by ${from}`);
}
showMessage("Hello~");

// 4. Rest parameters (added in ES6)
function printAll(...args) { /* ... -> 배열 형태로 전달 */
  for( let i = 0; i < args.length; i++) {
    console.log(args[i]);
  }

  /* 간단하게 출력하기 */
  for(const arg of args) { /* args의 값들이 하나씩 arg에 지정됨 */
    console.log(arg);
```

```
  }

  /* 더 간단하게 출력하기 */
  args.forEach((arg) => console.log(arg));
}
printAll("dream", "coding", "ellie");

// 5. Local scope
// 밖에서는 안이 보이지 않고 안에서만 밖을 볼 수 있다
let globalMessage = "global"; // global variable
function printMesseage() {
  let message = "hello";
  console.log(message); // local variable
  console.log(globalMessage);

  function printAnother() {
    /* 자식은 부모에게서 정의된 값을 확인 가능 */
    console.log(message);
    /* 자식안에 정의된 값은 부모(상위)에서 확인 불가능 */
    let childMessage = "hoho";
  }
  /* console.log(childMessage); */ /* -> 에러발생 */
  /* return underfined -> 이게 생략된거임 */
}
printMesseage();

// 6. Return a value
function sum(a, b) {
  return a + b;
}
const result = sum(1, 2);
console.log(`sum : ${sum(1,2)}`);

// 7. Early return, early exit

// bad logic
function upgradeUser(user) {
  if(user.point > 10) {
    // long upgrade logic...
    // 블럭안에서 로직을 많이 작성하면 가독성이 나빠짐
  }
}

// good logic
function upgradeUser(user) {
  if (user.point <= 10) {
    return; // 값이 아닐때는 빨리 종료 후 리턴값 반환
  }
  // long upgrade logic...
  // 조건이 맞을때만 긴 로직을 실행
}
// 8. Function expression

// First-class function
// 함수는 변수와 마찬가지
// functions are treated like any other variable
// 변수로 지정가능
// can be assigned as a value to variable
// 파라미터로 전달가능
// can be passed as an argument to other function
// 리턴값으로 전달가능
```

```
123 // can be returned by another function
124
125 // 1. Function expression
126 // 2. function declaration can be called earlier than it is defiend. (hoisted)
127 // function expression -> 할당된 다음부터 호출이 가능
128 // 3. function expression is created when the execution reached it.
129
130 // 함수 선언과 동시에 변수에 할당
131 const print = function () { // anonymous function
132   console.log("print");
133 };
134 print(); // 변수에 할당된 함수를 호출
135
136 const printAgain = print;
137 printAgain();
138 const sumAgain = sum;
139 console.log(`sumAgain : ${sumAgain(1, 5)}`);
140
141 // 9. Callback function using function expression
142 // 함수를 전달하여 상황에 맞으면(니가 원하면) 전달된 함수를 불러 -> callback function
143 // 두가지의 callback function이 parameter로 전달되었음
144 function randomQuiz(answer, printYes, printNo) {
145   if (answer === "love you") {
146     printYes();
147   } else {
148     printNo();
149   }
150 }
151
152 // anonymous function
153 const printYes = function () {
154   console.log("Yes!!!");
155 };
156 // named function
157 // better debugging in debugger's stack traces
158 // recursions -> 재귀
159 const printNo = function print () {
160   console.log("No!!!");
161   // print(); -> 계속해서 호출 숫자가 늘어남 / 피보나치 계산, 반복 평균값 계산등에서만 제한적
    사용
162 };
163 randomQuiz("wrong", printYes, printNo);
164 randomQuiz("love you", printYes, printNo);
165
166 // 10. Arrow function
167 // always anonymous
168 // 배열, 리스트등에서 활용도 높음
169 const simplePrint = function () {
170   console.log("simplePrint!!!");
171 };
172 const simplePrint2 = () => console.log("arrow function!!!");
173
174 const add = function (a, b) {
175   return a + b;
176 };
177 const add2 = (a, b) => a + b;
178
179 const simpleMultiply = (a, b) => {
180   // do something more
181   return a * b; // -> 블럭 사용시 리턴이 필수
182 };
```

```
183
184 // 11. IIFE : Immediately Invoked Function Expression
185 // 함수를 선언함과 동시에 호출 실시
186 (function hello() {
187   console.log("IIFE");
188 })();
```